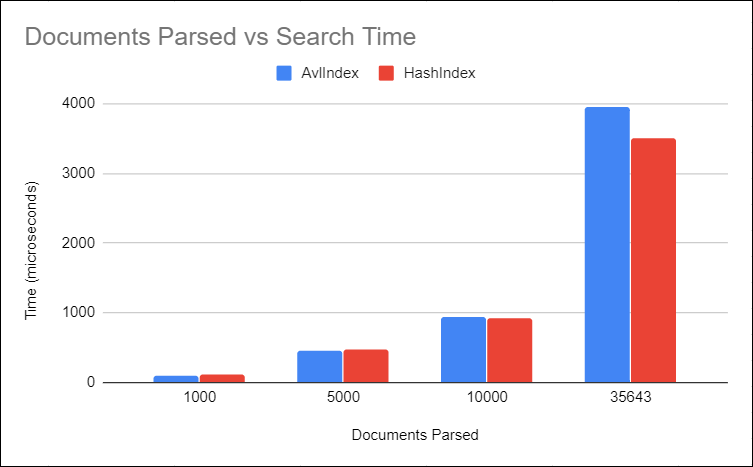
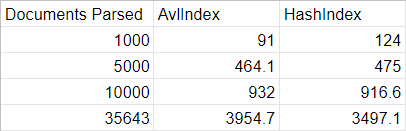
Zech Wolf: Search Engine Analysis

To test the effectiveness of searching with an AVL Tree versus a hash table, I set up four different data sets to search through. The first data set represented the entire corpus, and was loaded in from a persistent index. The other three sets were parsed at the time of testing, with sizes of 10k, 5k, and 1k documents. In order to control variables that could affect searching times, I ensured that the same files were parsed each time, and that the indexes contained the same words between each test.

Each test consisted of ten trials. For each trial, the populated index performed a search for the term “court” and I recorded its performance. I chose court because, according to the index statistics, it was the most frequently occurring word across all documents. After the ten trials were completed I recorded the timing information and constructed the following graphs and data sets.



The most noticeable result from the data is that the hash index is about 10% faster for a search on the full corpus. The reason for this, I believe, is that the average time complexity of searching an AVL tree is O(log n). As the number of elements in the index grows, the height of the tree grows, and the binary search becomes less efficient. However, the hash table can search with an average time complexity of O(1). But because the hash table used in this project handles collisions using separate chaining, it can be disadvantaged by the size of its individual chains. Although the hashing function can return a key for a requested value in constant time, there is a chance that the key will not be unique to the value. This means the chain will have to be searched linearly, reducing the efficiency of the search. However, this was not a major problem for the larger data set, and the disadvantages of the AVL tree ended up outweighing the disadvantages of the hash table. In my analysis, this is because the hashing algorithm I used was able to ensure that the chain length was insignificant in comparison to the size of the data set, keeping the average search time complexity close to O(1).

However, when the data set size was 1000, the AVL tree performed about 25% faster than the hash table. In my analysis, this is because the disadvantages of the hash table are amplified with a smaller set. With a smaller set, collisions occur in the hash table more frequently, thus the average length of the individual chains is longer. As a result, the length of each chain is significant in comparison to the size of the set, meaning the search will not be as close to constant time complexity. However, the AVL tree is able to perform a binary search in O(log n) time, which on average is faster than the hash table for 1000 documents.

In conclusion, the AVL tree and hash table perform very similarly at medium data set sizes (5k – 10k documents parsed). For small data sets, the AVL tree search is faster by about 25%, and for large data sets, the hash table is about 10% faster.